POCKET COMPUTER NEWSLETTER

© Copyright 1985 POCKET COMPUTER NEWSLETTER

Issue 39 - March/April

HAND-HELD COMMUNICATES VIA RADIO WAVES

Motorola's Communications Sector has introduced an industry first: a hand-held computer capable of communicating in real-time using radio waves in the 800 megahertz frequency range!

Dubbed the KDT Computer Terminal, the environmentally rugged unit weighs under 28 ounces and measures 7-1/2 by 4 by slightly over 1-1/4 inches. The unit is battery operated, contains an 800 megahertz data radio, built-in antenna, and sophisticated software. It also sports a 2-line display (liquid-crystal) capable of displaying 27 characters per line. The keyboard contains 59 keys and has full alphabetic capability arranged in QWERTY format, programmable function keys, and a numeric calculator pad.

Two microprocessors split duties to provide multi-functional capability. That is, the unit can support communication functions while it is simultaneously running a user's application program. The unit can accomodate up to 160 kilobytes of ROM and 80 kilobytes of RAM internally, with access to external memory and peripheral devices provided via serial or parallel I/O interfaces. User applications can reside in 32 kilobyte ROM modules which may be plugged into the unit. Up to four such application programs may be installed at one time.

The radio system design that supports the KDT device can manage more than a thousand users on a single radio channel. This is accomplished through multiplexing of the system. When utilizing the built-in data radio, communications occurs at a rate of 4800 bits per second over standard 25 or 12.5 kilohertz channels. If radio coverage is not available, the built-in modem operates over voice-grade phone lines at a 300 bits per second rate.

To obtain more information about the KDT Portable Data Communications System, contact: *Motorola, Inc., 1301 East Algonquin Road, Schaumburg, IL 60196*. Address your inquiries to the Communications Sector.

Photo *Motorola KDT 800 Portable Terminal.*

Another *Personal Information*™ product.

FOR HEWLETT-PACKARD HP-71B USERS

APPOINTMENTS PROGRAM WITH ALARM

A feature lacking on the HP-71B (that was provided on its predecessor, the HP-75) is an appointment calendar equipped with an audio alarm. The 71B is capable of providing this service from a hardware point of view, since it has built-in timers and an audible annunciator. But, Hewlett-Packard elected not to include the software routines that would provide such capability in with the basic unit. (After all, if they included *all* the features of the 75 at half the price, it would be a lot harder to appease the early 75 customers!)

The package of routines provided here yield much of the capability for scheduling appointments that is found in the HP-75. You can enter appointment dates and times (through the year 1999) and a reminder message. Appointments may be scheduled as repeating (at a cycle of anywhere between 1 minute to 24 hours) or as single events. You are reminded of appointments through an audible warning. Capability is provided to have an alarm issued even if the HP-71B has been turned off or is executing another program. (This capability is optional and is selected by the user as desired.) If this special capability is not elected, then a past-due appointment is normally announced whenever the HP-71B is turned on. Appointments are acknowledged and/or erased via keyboard control. With slight modification, you can utilize the basic package to have the HP-71B perform other programs when an alarm occurs.

Space considerations do not permit a detailed discussion of the routines. Thus, this article will deal primarily with operational instructions.

The routines are distributed over three files. The major file is named ALARMS. Two associated files are named PWRDNALM (Power Down Alarm) and CHECKALM (Check Alarm). The former is used if you want to have the HP-71B "wake-up" and sound an alarm when it is in the "off" mode. The latter is used whenever it is desired to see whether an appointment time has been reached or is past-due. (This routine can be periodically called by any other program to provide real-time alarm interrupt capability.) Finally, a fourth file named APPOINTS is used as a data file in which appointment information is maintained. (This file is created automatically by the ALARMS routine.) The APPOINTS file contains room for a maximum of 20 appointments. All together the various routines and the data file consume slightly more than 5K of memory.

As a check on your keyboarding accuracy, the amount of space consumed by each program file, as shown by the CAT command, should be:

ALARMS = 3201 bytes
PWRDNALM = 583 bytes
CHECKALM = 430 bytes

Program two "user" keys (using the DEF KEY command) such as the letters "a" and "p" to execute (RUN) the ALARMS and PWRDNALM programs when invoked. The CHECKALM routine should be invoked whenever the HP-71B is turned on. (Use the STARTUP command to "RUN CHECKALM".) It may also be called by any other program(s) if you wish to have alarms announced when you are running other packages. [Note: use the file names exactly as shown because the programs call one-another by those exact titles.]

To begin using the appointment scheduler, execute the ALARMS program. Respond to the initial ENTER/CANCEL query with the appropriate character depending on whether you wish to enter a new appointment or cancel an already existing entry. (Note that the package automatically provides the most common expected response to any prompt. When your response matches this, all you need do is press the END LINE key.)

Respond appropriately to further prompts. When entering an alarm date/time, just edit the current date/time that is provided with the prompt. Note that time is inputted using a 24-hour format. (Also note that you cannot enter an appointment whose date/time is prior to the current date/time.) Additional prompts allow you to direct that an alarm update and repeat at the interval you specify (minimum of 1 minute, maximum of 24 hours) and to enter a message indicating the nature of the appointment. (This reminder message may be up to 22 characters in length.)

Each time an appointment is entered, it is ordered in the APPOINTS file by date/time of occurrence. Thus, at the end of each entry procedure, you will observe a brief "sorting" message on the display. At the conclusion of each entry, the ALARMS program exits by running CHECKALM to see whether any alarms are due. The program then ends with the HP-71B in the command mode. To enter another alarm, you must RUN the ALARMS program again. Hence, the earlier suggestion that you program a user key with this directive.

If you want to be reminded of a pending appointment when the HP-71B is turned off, execute the PWRDNALM module. The HP-71B will go to sleep, but continue checking for the next appointment. Should one occur, the unit "awakens" and sounds the alarm in the "loud" mode. You "acknowledge" an alarm (stop the audible indicator) by pressing any key. Once an alarm has been "acknowledged" you may use the cursor keys to view any appointment reminder message. Pressing the END LINE key after an alarm has been acknowledged results in the following.

If the alarm is a repeating one, its new time will automatically be calculated.

Then, the user is asked whether or not to cancel the alarm. If it is not cancelled, the alarm is maintained in the APPOINTS file. (Note that if it is a non-repeating alarm, it will remain as a "past-due" alarm.)

The user then has the option of viewing any other items in the APPOINTS file.

If the PWRDNALM program sounds an alarm that is not acknowledged within approximately 15 seconds, then the unit simply goes back to "sleep" and ignores the "past-due" alarm condition. Thus, the audio annunciator feature is a one-shot deal when using the PWRDNALM routine. This procedure saves a lot of wear and tear on the batteries in the event that you leave town for a few days and forget about an upcoming appointment stored in the 71B!

Regardless of whether you elect to use the PWRDNALM routine, the CHECKALM routine will be executed any time you power up your HP-71B. (Provided that you program the STARTUP statement to 'RUN CHECKALM'.) If this routine finds a past-due appointment condition, it will sound the audible alarm (in the quieter mode, since someone is obviously near the unit). The alarm can then be acknowledged and the appointment removed from the file, as desired.

Take note of the fact that the system is designed to alert you to the "oldest" past-due appointment in the file named APPOINTS. You must remove it (by cancelling the entry) before a later appointment will be serviced by the audio alarm system. (In summary, the APPOINTS file is kept in chronological order. The CHECKALM routine tests the first entry in the file.)

With a little study of the routines, which are modular by design, you can undoubtably find ways to customize or adapt them to suit your particular needs. For instance, you might elect to have one of your own subprograms executed whenever an alarm went off.

Program *ALARMS.*

```
10 DELAY 1,.5 @ WIDTH 96 @ DESTROY ALL
20 INPUT "ENTER/CANCEL (E/C)? ","E";W$
30 IF UPRC$(W$[1])#"E" AND UPRC$(W$[1])#"C" THEN BEEP @ GOTO 20
40 IF UPRC$(W$[1])="C" THEN 200
50 REM  (C) COPYRIGHT 1985
60 REM  SCELBI C.C., INC.
70 REM  ALL RIGHTS RESERVED
100 GOSUB 1000
110 GOSUB 1100
120 GOSUB 1200
130 GOSUB 1300
140 GOSUB 1400
150 CALL SORT
160 GOTO 210
200 CALL CANCEL
210 DISP "ALARM(S) PROCESSED."
220 RUN "CHECKALM"
1000 W$=DATE$ @ W$=W$[4,8]&"/"&W$[1,2]
1005 ON ERROR GOTO 1095 @ W$=W$[1,8] @ INPUT "ALARM DATE? ",W$;W$
1010 Y=VAL(W$[7,8]) @ IF Y<84 OR Y>99 THEN 1095
1015 M=VAL(W$[1,2]) @ IF M<1 OR M>12 THEN 1095
1020 IF M#2 THEN 1035
1025 IF Y/4=INT(Y/4) THEN D1=29 ELSE D1=28
1030 GOTO 1040
1035 IF M=4 OR M=6 OR M=9 OR M=11 THEN D1=30 ELSE D1=31
1040 D=VAL(W$[4,5]) @ IF D<1 OR D>D1 THEN 1095
1045 W$=W$[7,8]&"/"&W$[1,5] @ IF W$<DATE$ THEN 1095
1050 D$=W$ @ OFF ERROR @ RETURN
1095 BEEP @ DISP "INVALID DATE." @ GOTO 1000
1100 W$=TIME$[1,5] @ ON ERROR GOTO 1195
1105 INPUT "ALARM TIME? ",W$;W$ @ W$=W$[1,5]
1110 Y=VAL(W$[1,2]) @ IF Y<0 OR Y>23 THEN 1195
1115 M=VAL(W$[4,5]) @ IF M<0 OR M>59 THEN 1195
1120 IF D$>DATE$ THEN 1130
1125 IF TIME$[1,5]>W$ THEN 1195
1130 T$=W$ @ OFF ERROR @ RETURN
1195 BEEP @ DISP "INVALID TIME." @ GOTO 1100
1200 W$="" @ ON ERROR GOTO 1295
1205 INPUT "REPEAT ALARM (Y/N)? ","N";W$
1210 IF UPRC$(W$[1])#"Y" THEN 1260
1215 INPUT "INTERVAL? ","24:00";W$ @ W$=W$[1,5]
1220 D=0
1225 D=D+1 @ IF D=3 THEN 1225
1230 IF NUM(W$[D])<48 OR NUM(W$[D])>57 THEN 1295
1235 IF D<5 THEN 1225
1240 Y=VAL(W$[1,2]) @ IF Y<0 OR Y>24 THEN 1295
1245 M=VAL(W$[4,5]) @ IF M<0 OR M>59 THEN 1295
1250 IF Y=0 AND M=0 THEN 1295
1255 IF Y=24 AND M#0 THEN 1295
1260 I$=W$ @ IF I$="N" THEN I$=" "
1265 OFF ERROR @ RETURN
1295 BEEP @ DISP "INVALID INTERVAL." @ GOTO 1200
1300 ON ERROR GOTO 1395 @ W$=""
1305 INPUT "ALARM MESSAGE? ",W$;W$
1310 IF LEN(W$)>22 THEN 1395
1315 M$=W$ @ OFF ERROR @ RETURN
1395 BEEP @ DISP "MESSAGE TOO LONG." @ GOTO 1300
1400 ON ERROR GOTO 1485 @ Y=0
1405 Y=Y+1
1410 W$=CAT$(Y)[1,8]
1415 IF W$="APPOINTS" THEN 1445
1420 IF W$#"" THEN 1405
1425 CREATE DATA APPOINTS,20,52
1430 ASSIGN #1 TO APPOINTS
1435 FOR Y=0 TO 19 @ PRINT #1,Y;"" @ NEXT Y
1440 ASSIGN #1 TO *
1445 ASSIGN #1 TO APPOINTS
1450 Y=-1
1455 Y=Y+1 @ IF Y>19 THEN ASSIGN #1 TO * @ GOTO 1485
1460 READ #1,Y;W$
1465 IF W$#"" THEN 1455
1470 PRINT #1,Y;D$,T$,I$,M$
1475 ASSIGN #1 TO *
1480 GOTO 1495
1485 DISP "APPOINTS FILE ERROR."
1490 DISP "APPOINTMENT CANCELLED."
1495 OFF ERROR @ RETURN
1500 SUB SORT @ GOSUB 1580 @ IF M<2 THEN 1595
1505 ASSIGN #1 TO APPOINTS @ DISP "SORTING APPOINTMENTS."
1510 FOR Y=1 TO M-1
1515 FOR D=Y+1 TO M
1520 READ #1,Y-1;D$,T$,I$,M$
1525 READ #1,D-1;D1$,T1$,I1$,M1$
1530 W$=D$&T$ @ W1$=D1$&T1$
1535 IF W$<=W1$ THEN 1550
1540 PRINT #1,Y-1;D1$,T1$,I1$,M1$
1545 PRINT #1,D-1;D$,T$,I$,M$
1550 NEXT D @ NEXT Y
1555 ASSIGN #1 TO *
1560 GOTO 1595
1580 ASSIGN #1 TO APPOINTS @ M=-1
1585 M=M+1 @ READ #1,M;W$ @ IF W$#"" THEN 1585
1590 ASSIGN #1 TO * @ RETURN
1595 END SUB
1600 SUB CANCEL @ ASSIGN #1 TO APPOINTS @ M=0 @ DIM W$[50]
1605 READ #1,M;D$ @ IF D$="" THEN 1795
1610 READ #1,M;D$,T$,I$,M$
1615 W$=D$[4,8]&"/"&D$[1,2]&" "&T$&" "&I$&" "&M$
1620 DELAY 8,8 @ DISP W$ @ W$=KEY$
1625 INPUT "CANCEL (Y/N)? ","N";W$
1630 IF UPRC$(W$[1])="Y" THEN GOSUB 1700 @ M=M-1
1635 M=M+1 @ IF M>19 THEN 1795
1640 INPUT "LIST NEXT (Y/N)? ","Y";W$
1645 IF UPRC$(W$[1])="Y" THEN 1605 ELSE 1795
1700 D=M+1
1705 READ #1,D;W$ @ IF W$="" THEN 1720
1710 READ #1,D;D$,T$,I$,M$
1715 PRINT #1,D-1;D$,T$,I$,M$ @ D=D+1 @ IF D<20 THEN 1705
1720 PRINT #1,D-1;"" @ RETURN
1795 ASSIGN #1 TO * @ DELAY 1,.5 @ END SUB
1800 SUB UPDATE(D$,I$,M$,T$) @ Y1=VAL(T$[1,2])+VAL(I$[1,2])
1805 M1=VAL(T$[4,5])+VAL(I$[4,5])
1810 IF M1>=60 THEN M1=M1-60 @ Y1=Y1+1
1815 IF Y1<24 THEN 1895
1820 Y1=Y1-24 @ D=D+1
1825 D=VAL(D$[7,8]) @ Y=VAL(D$[1,2]) @ M=VAL(D$[4,5])
1830 IF D=31 THEN M=M+1 @ D=1 @ GOTO 1855
1835 IF D=30 AND (M=4 OR M=6 OR M=9 OR M=11) THEN M=M+1 @ D=1 @ GOTO 1865
1840 IF D=29 AND M=2 AND Y/4=INT(Y/4) THEN M=M+1 @ D=1 @ GOTO 1865
1845 IF D=28 AND M=2 AND Y/4#INT(Y/4) THEN M=M+1 @ D=1 @ GOTO 1865
1850 D=D+1 @ GOTO 1865
1855 IF M>12 THEN M=1 @ Y=Y+1
1860 IF Y>99 THEN DISP "CALENDAR OVER-RANGE." @ STOP
1865 W$=STR$(Y) @ IF LEN(W$)<2 THEN W$="0"&W$
1870 D$=W$
1875 W$=STR$(M) @ IF LEN(W$)<2 THEN W$="0"&W$
1880 D$=D$&"/"&W$
1885 W$=STR$(D) @ IF LEN(W$)<2 THEN W$="0"&W$
1890 D$=D$&"/"&W$
1895 W$=STR$(Y1) @ IF LEN(W$)<2 THEN W$="0"&W$
1900 T$=W$
1905 W$=STR$(M1) @ IF LEN(W$)<2 THEN W$="0"&W$
1910 T$=T$&":"&W$
1915 ASSIGN #1 TO APPOINTS @ PRINT #1,0;D$,T$,I$,M$ @ ASSIGN #1 TO * @ END SUB
```

Program *CHECKALM.*

```
100 DESTROY ALL @ GOSUB 900 @ DIM W$[52] @ DELAY .05,8 @ IF D$="" THEN 195
105 IF D$>DATE$ THEN 195
110 IF D$<DATE$ THEN 120
115 IF T$>TIME$ THEN 195
120 OFF TIMER #1 @ W$=D$[4,8]&"/"&D$[1,2]&" "&T$&" "&I$&" "&M$ @ DISP W$
125 FOR Y=1 TO 60
130 BEEP 1000,.2 @ WAIT .05
135 W$=KEY$ @ IF W$#"" THEN Y=64
140 NEXT Y
145 IF Y<62 THEN 195
150 IF I$=" " THEN 190
155 CALL UPDATE(D$,I$,M$,T$) IN ALARMS
190 DELAY 8 @ CALL CANCEL IN ALARMS @ CALL SORT IN ALARMS
195 CFLAG ALL @ DELAY 1,.5 @ DISP CHR$(27)&"J";CHR$(27)&"E";">" @ END ALL
900 ASSIGN #1 TO APPOINTS
905 READ #1,0;D$ @ IF D$="" THEN RETURN
910 READ #1,0;D$,T$,I$,M$
915 ASSIGN #1 TO * @ RETURN
```

Program *PWRDWALM.*

```
10 ON TIMER #1,59 GOTO 50 @ D1=0
20 T1=TIME @ BYE
30 T2=TIME @ IF T2<59 THEN T2=86400-T1+T2 ELSE T2=T2-T1
40 IF T2<58 THEN CFLAG -25 @ RUN CHECKALM
50 IF D1#0 THEN 20
60 SFLAG -25
70 GOSUB 100
80 IF Y<62 THEN 20 ELSE 10
90 END
100 GOSUB 900 @ DIM W$[52] @ DELAY .05,8 @ IF D$="" THEN 230
110 IF D$>DATE$ THEN 230
120 IF D$<DATE$ THEN 140
130 IF T$>TIME$ THEN 230
140 D1=1 @ W$=D$[4,8]&"/"&D$[1,2]&" "&T$&" "&I$&" "&M$ @ DISP W$
150 FOR Y=1 TO 60
160 BEEP 1000,.2 @ WAIT .05
170 W$=KEY$ @ IF W$#"" THEN Y=64
180 NEXT Y
190 IF Y<62 THEN 230
200 IF I$=" " THEN 220
210 CALL UPDATE(D$,I$,M$,T$) IN ALARMS
220 DELAY 8 @ CALL CANCEL IN ALARMS @ CALL SORT IN ALARMS
230 CFLAG -25 @ DELAY 1,.5 @ RETURN
900 ASSIGN #1 TO APPOINTS
905 READ #1,0;D$ @ IF D$="" THEN RETURN
910 READ #1,0;D$,T$,I$,M$
915 ASSIGN #1 TO * @ RETURN
```

FOR PC-1250/51/60/61 USERS

MEMORY MAP FOR THE PC-1261

Merlin Rober, 407 North 1st Avenue, Marshalltown, IA 50158, is back with more vital information. This time he provides a memory map for the relatively new Sharp PC-1261. You may have to squint a bit to see everything as we reduced the material in order to squeeze as much information as possible into this edition. PC-1261 owners will definitely find any squinting worth the effort!

Table *PC-1261 Memory Map.*

The PC-1261 uses the same CPU as the earlier PC-1250. However, the 1261 runs a good deal faster than the 1250, partly because of the fact that the 1261 uses a "link byte" following each line number (as the PC-1500 does). Also, the FOR/NEXT loops have been speeded up significantly; execution of NEXT takes only about .0075 seconds, twice as fast as the PC-1500! However, most other BASIC operations run only about half as fast as they do in the PC-1500.

Another rather surprising feature of the 1261 is the speed at which it CLOADs and CSAVEs. Not counting the time for the "header", the 1261 loads a kilobyte in just 27 seconds. (Compare with the PC-1500's 73 seconds per kilobyte!)

The 1261, unlike the 1250, uses standard ASCII character codes. Its tokens are also different. Nevertheless, the 1261 can CLOAD programs that were written on (and CSAVEd by) the 1250! While loading such a program, the 1261 uses a table in ROM to translate each code as it is received. It also calculates and inserts a link byte in each program line. It must be kept rather busy at such times!

Athough not supported in the instruction manual, the PC-1261's BASIC includes PEEK, POKE, CALL, CSAVEM, and CLOADM.

For those interested in the internal organization of the PC-1261, a memory map follows. Although it contains a number of "gaps", it includes all major locations. Note that pointers are stored low byte first.

0000-01FF	ROM (Inaccessible by PEEK)
2000-203B	Display Buffer, left half of top line
203C	Mode: 01=RESERVE, 02=PRO, 08=RUN
203D	01=BUSY, 02=PRINT, 08=Katakana, 10=SML, 20=SHIFT, 40=DEF
203E-203F	Not used
2040-207B	Display Buffer, left half of bottom line
207C	01=DEG, 02=RAD, 04=GRAD, 20=ERROR, 40=Unused mark
207D-207F	Not used
2080-20BF	Pending operand stack, 8 floating-point registers
20C0-27FF	No memory
2800-283B	Display Buffer, right half of top line
283C-283D	Not used
283E-283F	Pointer to start of RESERVE Memory
2840-287B	Display Buffer, right half of bottom line
287C-287F	Not used
2880-288F	Pending-operation stack, 16 codes
2890-28B7	Arithmetic workspace
28B8-28BF	Current value of random number
28C0-3FFF	No memory
4000-407F	EASY SIMULATION Area (unexpanded)
4080-64FF	BASIC program area. FF byte in 4080; program begins 4081. Defined variables and arrays begin 64FF, working backward.
6500-6507	Z (Z$)
6508-650F	Y (Y$)
6510-6517	X (X$)
6518-651F	W (W$)
6520-6527	V (V$)
6528-652F	U (U$)
6530-6537	T (T$)
6538-653F	S (S$)
6540-6547	R (R$)
6548-654F	Q (Q$)
6550-6557	P (P$)
6558-655F	O (O$)
6560-6567	N (N$)
6568-656F	M (M$)
6570-6577	L (L$)
6578-657F	K (K$)
6580-6587	J (J$)
6588-658F	I (I$)
6590-6597	H (H$)
6598-659F	G (G$)
65A0-65A7	F (F$)
65A8-65AF	E (E$)
65B0-65B7	D (D$)
65B8-65BF	C (C$)
65C0-65C7	B (B$)
65C8-65CF	A (A$)
65D0-65FF	RESERVE Memory (48 bytes)
6600-6617	?
6618	Current horizontal position of display cursor
6619-661A	Pointer to start of editable (MERGEd) program
661B-661C	Pointer to start of program being executed
661D-6620	?
6621-6622	Pointer to start of EASY SIMULATION area
6623-6624	Pointer to end of EASY SIMULATION area
6625-6626	Pointer to last entry into EASY SIMULATION program
6627-6628	BCD idle-time counter, non-resetting
6629-662A	?
662B	Bit 4=display flash flag
662C	Code of last key used
662D	User-defined-key label sought
662E-6634	?
6635-6636	Pointer to INPUT or assignment statement
6637-6638	?
6639-663A	Pointer, BASIC locations
663B-663F	?
6640-6647	Stored Password
6648-664F	Floating-point accumulator, numeric data or string pointer
6650-667F	Buffer, for output to display
6680-6681	Auto power-off countdown, low byte first
6682	?
6683-6696	GOSUB stack (ten return addresses)
6697-66AF	Numeric results, formatted for display or printing
66B0-66B7	?
66B8-66CF	Buffer, for output to printer
66D0-66D6	?
66D7	Flags: 08=PRINT=LPRINT
66D8	Flags: 04=WAIT
66D9	Flags: 20=password-protected
66DA-66E0	?
66E1-66E2	Pointer to start of BASIC program
66E3-66E4	Pointer to end of BASIC program
66E5-66E6	WAIT time setting, low byte first
66E7	?
66E8	Code of character being flashed
66E9	Flashing cursor location
66EA	Pointer to Input Buffer, low byte
66EB-66ED	Pointer to variable used in assignment statement

66EE-66EF	Pointer to variable used in INPUT statement
66F0-66F2	?
66F3	Pointer to FOR/NEXT stack, low byte
66F4	Pointer to GOSUB stack, low byte
66F5	Pointer to pending operand stack, low byte
66F6	Pointer to pending operation stack, low byte
66F7	Pointer to String Buffer, low byte
66F8	USING, 08=scientific notation
66F9	USING, positions preceding decimal point
66FA	USING, POSITIONS following and including decimal point
66FB	USING, positions in character strings
66FC-66FD	Pointer to start of defined variables and arrays
66FE-66FF	Pointer to DATA item
6700-6701	Pointer to end of BASIC statement just executed
6702-6703	Pointer to BASIC at halt in execution
6704-6705	?
6706-675F	FOR/NEXT stack (5 loops, each consisting of 18 bytes)
6760-67AF	String Buffer, 80 bytes
67B0-67FF	Input Buffer, 80 bytes
7000-7FFF	Not used
8000-8044	ROM Begins: Jump instructions
8045-820B	Table of codes for generating display characters
820C-8391	Table, codes for Katakana, etc.
8392-839F	Codes for "BREAK IN" and "ERROR"
83A0-83D5	Key codes
83D6-840B	Key codes, Katakana mode
840C-8441	Key codes, SHIFTed
8442-8477	Key codes, Katakana mode, SHIFTed
8478-84AD	Key codes, SML mode
84AE-84E3	Key codes, SML mode, SHIFTed
84E4-84FD	Table of codes for keys A to Z, used with DEF key
84FE-A176	ROM routines
A177-A1AA	Table, address of token word, for given initial letter
A1AB-A254	Table, address of token word, for given token
A255-A4CE	Table of token words, tokens, and routine addresses
A4CF-A58D	Code conversion table for PC-1250 cassettes
A58E-D626	ML code, BASIC interpreter
D627-D63B	EASY SIMULATION messages "CLEAR", "YES-1 OR NO-2"
D63C-E7BD	ML Code
E7BE-E88F	Error messages
E890-E955	ASCII Codes, for display with "HELP"
E956-E98A	Table of addresses (by first letter) in "HELP" section
E98B-F456	"HELP" displays, involving BASIC keywords
F457-FFFF	Unused ROM, except possibly for a few short subroutines

MACHINE LANGUAGE UPDATE

Rick Wenger, 6221-18th Avenue, Kenosha, WI 53140, sends the following update on the information that appeared in Issue 38 of ***PCN***:

An exhaustive analysis of PC-1250 ROM code has turned up some interesting new data on the machine language instruction set.

The following instructions do not occur anywhere in ROM and should be considered unsupported:

1. LDA (PC) and LD(P) PC (opcodes 56 and 54).
2. The ZBIT and ZCP instruction (codes 76, 77, 72 & 73).
3. CLR (SP) (opcode D8).
4. CLA (opcode 23) and NOP (opcode 33).

The problem with using any of these instructions in machine language routines, is that if Sharp ever decides to upgrade this CPU, these instructions might go by the wayside. Serious software developers will want to stay clear of them. (They are primarily of interest only to hardcore CPU sleuths!)

The following opcodes do not occur in the PC-1250 ROM, but may be considered "valid" opcodes: 21, 4A, 4B, 9D, 9F, A3, AB, B2, B3, B6, BB, BC, BD, BE, C0, C5, C6, CA, CB, F4 and FD. All of these (except C0) were listed in Issue 38 of ***PCN***. All of these fit a pattern within the instruction set. It is understandable in each case, why the ROM programmer's never got around to utilizing them in the PC-1250. (Thus, there are two separate reasons for assuming that the opcodes represent "valid" operations.)

Twenty-three other opcodes were also not used in the ROM and were not listed as instructions in the previous article. Most of these opcodes will crash if an attempt is made to execute them in a program.

The following information serves as a correction or update to the material that appeared on pages 4 and 5 in Issue 38 of ***PCN***:

The explanation for ADECO (P) (P') occurs under the ADECO (P) A instruction.

The instruction ADECO (P) A (opcode 0C) adds the contents of the accumulator to the BCD number pointed to by P. P is decremented and then zero is added with carry. This adding of zero with carry is done the number of times indicated by the contents of C0. (P-C0-1 goes into P.)

The opcode for LD1 (P) (P') is 0A, not the DA shown.

The explanation under EXA (P) should read: Exchange contents of A and (P). (...etc...)

The mnemonic for the opcde 0B should read: EX1 (P)(P').

The LOAD P instructions (such as SLP 00) load register P with immediate data, not with an instruction.

The forward and reverse relative branches are computed by counting from the final byte of the instruction (the nn address), not from the initial byte of the next instruction.

The explanation for the RRO (P) instruction appears on the two lines right above it.

The opcode for INCC1 should be C0, not 60.

The opcode for INC (09) should be C8, not 68.

The opcode for INC (0B) should be CA, not 6A.

The opcode for DEB should be C3, not 63.

To clarify the effects of the increment and decrement instructions on the C and Z flags: C and Z are only affected by increments/decrements on 8-bit registers. Seven- and sixteen-bit increments/decrements do not affect the flags.

The INC IF C0 (opcode 6F) instruction is incorrect and should be deleted from the list.

There is more to come! The analysis made to date indicates there are 9 more valid opcodes available for use by this CPU. Look for a discussion of them in a future issue of ***PCN***.

FOR PC-1350 USERS

MEMORY MAP FOR PC-1350

See the article under the PC-1250/51 column that provides updates to the instruction set (as the same CPU is also used in the Sharp PC-1350).

In addition to his PC-1261 sleuthing work, *Norlin Rober, 407 N. 1st Avenue, Marshalltown, IA 50158,* has also been doing a lot of exploring on the Sharp PC-1350. He notes that, except for its four-line graphic display, the PC-1350 does not differ radically from the PC-1260/61. It uses the same CPU, and its BASIC uses the same tokens (with a few additions). The 1350 is also the first Sharp PC to provide the entire set of standard ASCII characters (although the 1261 came close).

Norlin also reminds readers that the 1350 is capable of loading programs that have been recorded on cassettes by the earlier Sharp PC-1211 (Radio Shack PC-1), Sharp 1250/51 (Radio Shack PC-3 & PC-3A), and Sharp PC-1260/61. Some modifications may be required to the programs themselves (after loading), especially if the programmer used non-approved programming tricks such as utilizing implied multiplication on the 1211 or 1250.

He has been digging out other information that will be passed along in future issues of ***PCN***. Meanwhile, enjoy the fruits of his recent labors by perusing the accompanying memory map for the PC-1350!

Table *PC-1350 Memory Map.*

```
0000-0823  ROM, ML Code.  RESET routine starts 0000
0824-08B3  Constants for computation of transcendental functions
08B4-1F04  ML Code
1F05-1FFF  Unused ROM
2000-5FFF  RAM card expansion area.
           8K Card, 4000-5FFF; 16K Card, 2000-5FFF
6000-602F  System Management Area (with no RAM card in use);
           6006-602F is a copy of 6F00-6F29
6030       BASIC Start Marker (FF byte)
6031-6C2F  BASIC area.  Program starts 6031; defined variables
           and arrays end 6C2F.
6C30-6C37  Z (Z$)
6C38-6C3F  Y (Y$)
6C40-6C47  X (X$)
6C48-6C4F  W (W$)
6C50-6C57  V (V$)
6C58-6C5F  U (U$)
6C60-6C67  T (T$)
6C68-6C6F  S (S$)
6C70-6C77  ▷ (R$)
6C78-6C7F  Q (Q$)
6C80-6C87  P (P$)
6C88-6C8F  O (O$)
6C90-6C97  N (N$)
6C98-6C9F  M (M$)
6CA0-6CA7  L (L$)
6CA8-6CAF  K (K$)
6CB0-6CB7  J (J$)
6CB8-6CBF  I (I$)
6CC0-6CC7  H (H$)
6CC8-6CCF  G (G$)
6CD0-6CD7  F (F$)
6CD8-6CDF  E (E$)
6CE0-6CE7  D (D$)
6CE8-6CEF  C (C$)
6CF0-6CF7  B (B$)
6CF8-6CFF  A (A$)
6D00-6D5F  Buffer, for output to display
6D60-6DFF  Additional buffer space, serial I/O
6E00-6E05  ?
6E06-6E5F  FOR/NEXT stack (Five 18-byte loop registers)
6E60-6EAF  String Buffer, 80 bytes
6EB0-6EFF  Input Buffer, 80 bytes
6F00       Flags
6F01-6F02  Pointer to start of BASIC program (low byte first)
6F03-6F04  Pointer to end of BASIC program
6F05-6F06  Pointer to start of editable (MERGEd) program
6F07-6F08  Pointer to start of defined variables and arrays
6F09-6F10  Stored Password
6F11       Flags: 02=Down-arrow-key execution
6F12       Flags: 08=PRINT=LPRINT
6F13       Flags: 04=WAIT
6F14       Flags: 20=Password-protected
6F15       Flags
6F16       Flags: 40=TEXT Mode; 80=Katakana Mode
6F17       Flags: 02=Display cursor enabled (except for INPUT);
           04=Display cursor enabled; 20=Redisplay expression
6F18-6F1B  ?
6F1C-6F1D  Pointer to start of program being executed
6F1E       Code for MEM$, "B" or "C"
6F1F       ?
6F20-6F21  Pointer to INPUT or assignment statement
6F22-6F23  Pointer to DATA item
6F24-6F26  Pointer to variable used in assignment statement
6F27-6F28  Pointer to variable used in INPUT statement
6F29       ?
6F2A       Pointer to Input Buffer address following INPUT prompt
6F2B       Pointer to FOR/NEXT stack, low byte
6F2C       Pointer to GOSUB stack ,low byte
6F2D       Pointer to pending operand stack, low byte
6F2E       Pointer to pending operation stack, low byte
6F2F       Pointer to String Buffer, low byte
6F30       ?
6F31       User-defined-key label being sought
6F32-6F35  ?
6F36       Code of character being flashed
6F37       ?
6F38       Pointer to Input Buffer, low byte
6F39       USING, 08=scientific notation
6F3A       USING, No. of positions preceding decimal point
6F3B       USING, No. of positions for decimal point and following
6F3C       USING, No. of positions for character strings
6F3D-6F3F  ?
6F40-6F43  GCURSOR coordinates
6F44-6F4B  ?
6F4C       Column for display of immediate-mode result
6F4D       ?
6F4E       Column for next LPRINT
6F4F       Pointer to Input Buffer during INPUT
6F50       ?
6F51-6F52  Codes for name and type of specified array
6F53-6F54  ?
6F55-6F56  Pointer to start of RESERVE memory
6F57       Code of last key used
6F58       ?
6F59-6F5B  JMP instruction, used by system M.L.
6F5C-6F5F  Coordinates of endpoint of previous LINE
6F60-6F61  ?
6F62       Flags for LINE, PSET: 01=set; 02=reset; 04=reverse;
           08=square; 20=specification for line type
6F63-6F66  Coordinates for SET, GPRINT, or start of LINE
6F67-6F6A  Coordinates for end point of LINE
6F6B-6F6E  ?
6F6F-6FFF  RESERVE Memory (6FFF reserved as end marker)
7000-701D  Display Buffer, first fifth of top row
701E-703B  Display Buffer, first fifth of 3rd row
703C-703F  Not used
7040-705D  Display Buffer, first fifth of 2nd row
705E-707B  Display Buffer, first fifth of 4th row
707C-707F  Not used
7080-7087  Floating-point accumulator (numeric or string pointer)
7088-708F  ?
7090-70A3  GOSUB Stack (ten return addresses)
70A4-70A5  Binary-coded-decimal idle-time counter, non-resetting
70A6       Key-repeat timing control
70A7       ?
70A8-70AF  Floating-point register, last calculated result
70B0-70B1  Auto power-off counter, low byte first
70B2       ?
70B3-70B4  WAIT time setting, low byte first
70B5-70B6  Pointer to RETURN statement (or Input Buffer)
70B7-70BB  ?
70BC-70BD  Pointer to VAL or STR$ in expression
70BE-70BF  ?
70C0-71FF  No memory
7200-721D  Display Buffer, 2nd fifth of top row
721E-723B  Display Buffer, 2nd fifth of 3rd row
723C-723F  Not used
7240-725D  Display Buffer, 2nd fifth of 2nd row
725E-727B  Display Buffer, 2nd fifth of 4th row
727C-727F  Not used
7280-7297  Character codes for result of immediate calculation
7298-729F  Current value of random number
72A0-72B8  Numeric results, formatted for display or printing
72B9-72BF  ?
72C0-73FF  No memory
7400-741D  Display Buffer, 3rd fifth of top row
741E-743B  Display Buffer, 3rd fifth of 3rd row
743C-743F  Not used
7440-745D  Display Buffer, 3rd fifth of 2nd row
745E-747B  Display Buffer, 3rd fifth of 4th row
747C-747F  Not used
7480-7497  Buffer, for output to printer
7498-7499  Pointer to end of BASIC statement just executed
749A-749B  Pointer to BASIC at halt in execution
749C-749D  Pointer to error location in BASIC program
749E-749F  ?
74A0-74AF  Pending-operation stack, 16 codes
74B0-74BF  ?
74C0-75FF  No memory
7600-761D  Display Buffer, 4th fifth of top row
761E-763B  Display Buffer, 4th fifth of 3rd row
763C-763F  Not used
7640-765D  Display Buffer, 4th fifth of 2nd row
765E-767B  Display Buffer, 4th fifth of 4th row
767C-767F  Not used
7680-76BF  Pending operand stack, 8 floating-point registers
76C0-77FF  No memory
7800-781D  Display Buffer, last fifth of top row
781E-783B  Display Buffer, last fifth of 3rd row
783C       Flags: 01=SHIFT; 02=DEF; 04=PRINT; 08=RESERVE; 10=RUN;
           20=PRO; 40=Katakana; 80=SML
783D       Flag
783E-783F  Not used
7840-785D  Display Buffer, last fifth of 2nd row
785E-787B  Display Buffer, last fifth of 4th row
787C       Flags: 01=DEGREE; 02=RADIAN; 04=GRAD
787D-787F  Not used
7880-7881  Row and column of flashing cursor
7882-788A  Display control
788B-788C  CURSOR coordinates
788D-788E  Pointer to Display Buffer
788F-789F  Display control
78A0-78AF  ?
7880       Serial I/O flag: 80=OPEN
78B1       Code for end of text
7882       Baud rate: 01=300, 02=600, 04=1200
7883       Endcode: 01=F, 02=L, 04=C; 10=2 stopbits;
           Parity: 40=O, 60=E; 80=8-bit word
78B4-78B5  Console specification
78B6-78BF  ?
78C0-7FFF  No memory
8000-8005  ROM; filled with 01
8006-8089  ML Jump instructions
808A-8269  Table: codes generating display characters
826A-83F4  Table: codes for Katakana characters
83F5-8402  Codes for "BREAK IN" and "ERROR"
8403-8443  Key codes
8444-8484  Key codes, Katakana mode
8485-84C5  Key codes, SHIFTed
84C6-8506  Key codes, Katakana mode, SHIFTed
8507-8547  Key codes, SML mode
8548-8588  Key codes, SML mode, SHIFTed
8589-85A2  Table: codes for A to Z when used with DEF key
85A3-A547  ML code: operating system, cassette routines
A548-A57B  Table: address of token word, for given initial letter
A57C-A637  Table: address of token word, for given token
A638-A91B  Table: token words, tokens, and routine addresses
A91F-A9DE  Code conversion table for PC-1250 cassettes
A9DF-AE9B  ML code
AE9C-AEBF  Codes for "RESERVE MODE", "PROGRAM MODE", "RUN MODE"
AEC0-FF82  ML Code: Routines for BASIC commands and statements;
           Display, printer, graphics, and serial I/O
FF83-FFFF  Unused ROM (filled with code CE)
```

FOR PC-1500 & PC-2 USERS

Program *PC! Solve.*

```
38D0 F2 E9 78 50
38D4 00 BE D0 2B
38D8 E9 78 80 00
38DC E9 76 4E FE
38E0 FD 58 B5 0A
38E4 FD CA FD 6A
38E8 4A 12 8E 12
38EC 20 20 20 20
38F0 20 20 20 20
38F4 50 43 21 20
38F8 53 4F 4C 56
38FC 45 52 BE ED
3900 3B BE E2 43
3904 C3 42 B7 0D
3908 9B 09 FD C8
390C BE D0 2B B5
3910 40 AE 78 80
3914 FD 8A 8E 08
3918 BE E8 CA BE
391C E2 43 C3 42
3920 EB 7B 0E 40
3924 B7 08 89 15
3928 5E B0 9B 14
392C 5E B1 89 03
3930 56 9E 1B 56
3934 56 15 B7 E0
3938 93 22 54 9E
393C 25 B7 0C 89
3940 0D 15 B7 0D
3944 9B 2B B7 E0
3948 81 01 54 54
394C 9E 36 E9 7B
3950 0E BF B7 1C
3954 89 05 BE CD
3958 E6 9E 43 B7
395C 1D 89 0D 15
3960 B7 E0 81 03
3964 BE CE 38 BE
3968 CE 38 9E 54
396C B7 0D 8B 25
3970 B7 1A 89 05
3974 BE D0 2B 9E
3978 61 B7 20 91
397C 62 5E FF 9B
3980 66 FD C8 15
3984 B7 E0 81 08
3988 FD 8A 51 BE
398C CE 38 9E 78
3990 FD 8A 51 9E
3994 7D EB 76 4E
3998 01 BE F9 57
399C BE CC DE 68
39A0 7B 6A FF B5
39A4 58 27 8B 0C
39A8 66 6E B0 99
39AC 08 EB 78 50
39B0 FF BA 3C 02
39B4 68 7B 6A FF
39B8 B5 2C 27 8B
39BC 04 88 05 8E
39C0 14 64 B5 50
39C4 27 8B 0F FD
39C8 28 FD 5A DE
39CC 09 48 79 4A
39D0 A8 BE DC 0C
39D4 8E 08 EC 48
39D8 79 4A A8 8E
39DC 02 9E C7 BE
39E0 DC 0C EC B5
39E4 FC AE 7A 00
39E8 B5 10 AE 7A
39EC 02 48 79 4A
39F0 00 BE DC 0C
39F4 EC B5 FF AE
39F8 7A 00 B5 50
39FC AE 7A 02 48
3A00 79 4A B0 BE
3A04 DC 0C 48 79
3A08 4A A8 BE DC
3A0C 20 48 79 4A
3A10 30 BE DC 0C
3A14 48 79 4A 28
3A18 BE DC 0C EC
3A1C 48 79 4A C8
3A20 BE DC 0C EC
3A24 E6 48 79 4A
3A28 C8 BE DC 20
3A2C B5 04 BE D0
3A30 D2 89 26 8E
3A34 02 9E 5A 48
3A38 79 4A B0 BE
3A3C DC 20 E6 48
3A40 79 4A 30 BE
3A44 DC 20 BE EF
3A48 B6 48 79 4A
3A4C 30 BE DC 0C
3A50 48 79 4A 10
3A54 BE DC 0C 8E
3A58 1B 48 79 4A
3A5C 30 BE DC 20
3A60 48 79 4A 10
3A64 BE DC 0C CD
3A68 54 CD 6A CD
3A6C 56 48 79 4A
3A70 C8 BE DC 0C
3A74 BE 3A 9A 48
3A78 79 4A B0 BE
3A7C DC 20 E6 48
3A80 79 4A 28 BE
3A84 DC 20 F0 48
3A88 79 4A 28 BE
3A8C DC 0C 48 79
3A90 4A 10 BE DC
3A94 0C BE 3A 9A
3A98 9E 41 CD A6
3A9C C9 42 48 79
3AA0 4A 10 BE DC
3AA4 20 48 79 4A
3AA8 B8 BE DC 0C
3AAC BE 3B A1 48
3AB0 79 4A 08 BE
3AB4 DC 20 48 79
3AB8 4A C0 BE DC
3ABC 0C 48 79 4A
3AC0 10 BE DC 20
3AC4 E6 48 79 4A
3AC8 00 BE DC 20
3ACC F0 48 79 4A
3AD0 B8 BE DC 0C
3AD4 BE 3B A1 48
3AD8 79 4A 10 BE
3ADC DC 20 48 79
3AE0 4A 18 BE DC
3AE4 0C 48 79 4A
3AE8 C0 BE DC 20
3AEC E6 48 79 4A
3AF0 00 BE DC 20
3AF4 CD 7E BE DB
3AF8 F5 48 79 4A
3AFC C0 BE DC 20
3B00 E6 48 79 4A
3B04 08 BE DC 20
3B08 BE EF B6 E6
3B0C CD 30 CD 58
3B10 83 9D E6 48
3B14 79 4A 18 BE
3B18 DC 20 BE EF
3B1C B6 48 79 4A
3B20 10 BE DC 0C
3B24 48 79 4A 10
3B28 BE DC 20 E6
3B2C 48 79 4A 18
3B30 BE DC 20 BE
3B34 EF B6 BE F5
3B38 97 E6 EC B5
3B3C F8 AE 7A 00
3B40 B5 10 AE 7A
3B44 02 CD 54 CD
3B48 66 B5 06 BE
3B4C D0 D2 99 B6
3B50 48 79 4A 10
3B54 BE DC 20 BE
3B58 F5 BE E6 48
3B5C 79 4A 10 BE
3B60 DC 20 BE EF
3B64 B6 CD 54 BE
3B68 F7 53 B5 F8
3B6C AE 7A 10 B5
3B70 10 AE 7A 12
3B74 B5 02 BE D0
3B78 D2 8B 09 48
3B7C 79 4A 10 BE
3B80 DC 20 8E 8D
3B84 48 79 4A 10
3B88 BE DC 20 E9
3B8C 76 4E FE B5
3B90 20 AE 78 80
3B94 BE E8 CA BE
3B98 E2 43 C3 42
3B9C EB 76 4E 01
3BA0 9A 58 7B 5A
3BA4 B0 DE 08 48
3BA8 79 4A 08 BE
3BAC DC 0C 9A 58
3BB0 7B 5A B0 BA
3BB4 39 18 48 79
3BB8 4A 10 BE DC
3BBC 20 E6 BE F5
3BC0 BE CD 54 CD
3BC4 66 BE EF B6
3BC8 E6 EC B5 F8
3BCC AE 7A 00 B5
3BD0 10 AE 7A 02
3BD4 CD 54 CD 66
3BD8 B5 02 BE D0
3BDC D2 89 0A 48
3BE0 79 4A 10 BE
3BE4 DC 20 BA 3C
3BE8 11 E9 76 4E
3BEC FE 48 79 4A
3BF0 10 BE DC 20
3BF4 B5 20 AE 78
3BF8 80 BE E8 CA
3BFC BE E2 43 C3
3C00 42 9A 58 7B
3C04 5A B0 DE 02
3C08 8E 06 58 7B
3C0C 5A B0 BA 39
3C10 18 48 79 4A
3C14 60 BE DC 0C
3C18 48 79 4A 60
3C1C BE DC 20 BE
3C20 F5 BE 48 79
3C24 4A 68 BE DC
3C28 0C CD 54 CD
3C2C 6A CD 56 48
3C30 79 4A 70 BE
3C34 DC 0C 48 79
3C38 4A 90 BE DC
3C3C 0C 48 79 4A
3C40 98 BE DC 0C
3C44 EC 48 79 4A
3C48 88 BE DC 0C
3C4C 48 79 4A A0
3C50 BE DC 0C 48
3C54 79 4A 68 BE
3C58 DC 20 E6 48
3C5C 79 4A 60 BE
3C60 DC 20 BE EF
3C64 B6 48 79 4A
3C68 78 BE DC 0C
3C6C 48 79 4A 98
3C70 BE DC 20 E6
3C74 48 79 4A 68
3C78 BE DC 20 CD
3C7C 7E E6 48 79
3C80 4A 88 BE DC
3C84 20 F0 48 79
3C88 4A A8 BE DC
3C8C 0C 48 79 4A
3C90 A0 BE DC 20
3C94 E6 48 79 4A
3C98 68 BE DC 20
3C9C CD 7E E6 48
3CA0 79 4A 90 BE
3CA4 DC 20 F0 48
3CA8 79 4A B0 BE
3CAC DC 0C 48 79
3CB0 4A B0 BE DC
3CB4 20 E6 48 79
3CB8 4A A8 BE DC
3CBC 20 CD 58 E6
3CC0 48 79 4A 60
3CC4 BE DC 20 CD
3CC8 54 CD 66 BE
3CCC EF B6 48 79
3CD0 4A 58 BE DC
3CD4 0C EC B5 F8
3CD8 AE 7A 00 B5
3CDC 10 AE 7A 02
3CE0 E6 48 79 4A
3CE4 58 BE DC 20
3CE8 BE F5 97 B5
3CEC 01 BE D0 D2
3CF0 89 99 CD A6
3CF4 C9 42 48 79
3CF8 4A 78 BE DC
3CFC 20 E6 48 79
3D00 4A 70 BE DC
3D04 20 CD 58 BE
3D08 F5 BE 48 79
3D0C 4A 68 BE DC
3D10 0C 48 79 4A
3D14 78 BE DC 20
3D18 48 79 4A 48
3D1C BE DC 0C 48
3D20 79 4A 78 BE
3D24 DC 20 E6 48
3D28 79 4A 68 BE
3D2C DC 20 CD 7E
3D30 E6 48 79 4A
3D34 70 BE DC 20
3D38 BE EF B6 48
3D3C 79 4A 78 BE
3D40 DC 0C 48 79
3D44 4A 48 BE DC
3D48 20 48 79 4A
3D4C 70 BE DC 0C
3D50 48 79 4A 98
3D54 BE DC 20 48
3D58 79 4A 88 BE
3D5C DC 0C 48 79
3D60 4A A8 BE DC
3D64 20 48 79 4A
3D68 98 BE DC 0C
3D6C 48 79 4A A0
3D70 BE DC 20 48
3D74 79 4A 90 BE
3D78 DC 0C 48 79
3D7C 4A B0 BE DC
3D80 20 48 79 4A
3D84 A0 BE DC 0C
3D88 BA 3C 6C F2
3D8C B5 20 AE 78
3D90 80 48 79 4A
3D94 B0 BE DC 20
3D98 E6 48 79 4A
3D9C A8 BE DC 20
3DA0 CD 58 BE E8
3DA4 CA E9 78 75
3DA8 00 E9 76 4E
3DAC FE 48 79 4A
3DB0 A8 BE DC 20
3DB4 BE D9 CF DC
3DB8 24 FD 6A CD
3DBC 92 B5 2F BE
3DC0 ED 4D 48 79
3DC4 4A B0 BE DC
3DC8 20 BE D9 CF
3DCC DC 24 FD 6A
3DD0 CD 92 BE E2
3DD4 43 C3 42 A5
3DD8 78 50 EB 76
3DDC 4E 01 B7 FF
3DE0 8B 01 9A E9
3DE4 76 4E FE B5
3DE8 40 AE 78 80
3DEC BE D0 2B E9
3DF0 78 50 00 BA
3DF4 39 18
```

PC! SOLVE

Eric Bowman, PEA Box 81, Exeter, NH 03833, submitted this program that he calls: PC! Solve. Here is how he describes its use and operation:

PC! Solve is a versatile equation solving and fractions program. It is written entirely in machine language, thus it is fast. The program can evaluate an expression and display the result as a decimal and fraction or solve an equation with real roots for an unknown.

Preparing the Program for Utilization

First, initialize memory in your PC by entering the command NEW &3DF6. Then, load the program. If loading from the accompanying listing, use POKE statements or utilize a Monitor program. Note that the program is *not* relocatable. Once loaded, start the program by executing: CALL &3800. When started, the program will display "PC! SOLVE" as an indication that it is ready to go. The next step is to enter an equation to be solved or an expression to be evaluated.

Solving An Equation

To solve an equation, enter the equation via the keyboard. All BASIC syntax is applicable here. You must represent the unknown by the symbolic variable "X" or the expression will be evaluated instead of solved. Note that the program is designed to solve for a root so the equation must be set equal to the value zero. Thus, to find the root of 4X=16, you must input 4*X-16 and then press the ENTER key.

If, in fact, you try the above (4*X-16), in about half a second you should see the number 4 appear on the display as 4 is the solution to that simple example.

The solve routine also has the capability for accepting user-defined starting points. If a function is suffixed with ",n" the routine will begin searching for a root at "n" along the X-axis. This capability is particularly valuable if the function to be solved takes the logarithm of X or divides by X or a multiple of X. For instance, to solve 3/X=6, you could input: 3/X-6,1. This causes the search for the root to begin at X=1 (instead of at X=0, which would result in an error). After a root is displayed, the search will continue for another case upon depression of any key except BREAK.

If a root is not an integer, it will be displayed as a fraction (reduced to lowest terms) and a decimal.

The method used to solve equations in this routine is known as Newton's Method. Note that any root located is only accurate to plus/minus 0.00000001. Therefore, this method is only an approximation of the root. For many practical purposes, however, this method is quite adequate.

Converting To A Fraction

If the letter "X" is not used when entering an expression, the expression will be evaluated once. The result will be displayed in both fractional and decimal formats. For instance, to find the fractional form of 0.111111111 . . ., the user would type: 0.11111111 and press the ENTER key. In about half a second, "1/9" will be displayed. Alternately, the user could elect to evaluate an expression such as this: ASN ((1/SQR(2))^2. Pressing the ENTER key following the inputting of these terms would result in "1/2" being shown.

Accuracy is again to 0.00000001. In the first example, if 0.1111111 was inputted (seven 1's after the decimal instead of eight), the result would not be "1/9" but rather 1111111/100000000. For repeating decimals, the entry accuracy should be to 10^(-8).

Errors

If there is an error, either in the solving process or in the initial evaluation of an expression, the inputted data will be re-displayed with the cursor flashing over the first character. If you are trying to solve, look for trouble spots such as division by zero or the extraction of a logarithm of a negative number. When the expression has been corrected, press ENTER to restart the process. Note that the editing features normally found when using BASIC will be in effect during such correction operations. If you want to erase the entire entry, type SHIFT/CL. (Note that as a safety precaution, use of the CL key alone has no affect when in this correction mode.)

Eric indicates that he welcomes inquiries from readers that have any questions or problems using the program or who have suggestions on improving the program.

Available Only by Prepaid Subscription for a Calendar Year Period (January - December). You are sent back issues for the calendar year to which you subscribe, at the time you enroll.

__ Enroll me as a 1985 Subscriber (Issue numbers 37-44). $24.00 in U.S. ($30.00 to Canada/Mexico. Elsewhere U.S. $40.00 payable in U.S. funds against a U.S. bank.)
__ Enroll me as a 1984-85 Subscriber (Issue numbers 31-44). $42.00 in U.S. ($51.00 to Canada/Mexico. Elsewhere U.S. $70.00 payable in U.S. funds against a U.S. bank.)
__ Enroll me as a 1983-85 Subscriber (Issue numbers 21-44). $78.00 in U.S. ($93.00 to Canada/Mexico. Elsewhere U.S. $120.00 payable in U.S. funds against a U.S. bank.)
__ Enroll me as a 1982-85 Subscriber (Issue numbers 11-44). $102.00 in U.S. ($125.00 to Canada/Mexico. Elsewhere U.S. $160.00 payable in U.S. funds against a U.S. bank.)
__ Check here if paying by MasterCard or VISA. Please give credit card information below.

Orders must be accompanied by payment in full. All checks must be magnetically encoded, payable in U.S. funds and drawn against a U.S. bank.

Name: _____

Addr: _____

City: _____ State: ____ Zip: _____

MC/VISA #: _____

Signature: _____ Exp. Date: _____

mail this order form to:

POCKET COMPUTER NEWSLETTER

P.O. Box 232, Seymour, CT 06483

FROM THE WATCH POCKET

If you ever wished you could have a *lot* more memory in your HP-71B computer, you should know that now you can. An outfit called **Hand Held Products, Inc.**, *P.O. Box 2388, Charlotte, NC 28211,* has announced a line of products that allow you to add as much as 96K (wow!) of memory to the -71B. You can also mix combinations of RAM and ROM. The expansion memory mounts in place of the magnetic strip reader module. Prices are pretty stiff -- it will cost you $995.00 to purchase the 96K module -- but if you are, say, developing programs in Forth or working in a commercial environment, you might find the extra memory well worth the price. Write to the company to obtain literature on these memory expansion products.

Word is floating around that Sharp is close to releasing a model called the PC-2500. The story goes that it will have a multiple-line display, a built-in 4-inch wide combination plotter/printer, and a keyboard large enough to touch-type on. (Sounds like a lap-sized portable!) One other tid-bit: the CPU will be the same one used in the 1250/1260/1350! Sure hope this turns out to be true.